just an integer. You might know that 0xB8000000 is the combined segment-offset address of video memory on your system, but nothing in the statement tells the program that this number is an address. C prior to the new C99 standard let you make assignments like this. But C++ more stringently enforces type agreement, and the compiler will give you an error message saying you have a type mismatch. If you want to use a numeric value as an address, you should use a type cast to convert the number to the appropriate address type:

```c
int * pt;
pt = (int *) 0xB8000000; // types now match
```

Now both sides of the assignment statement represent addresses of integers, so the assignment is valid. Note that just because it is the address of a type `int` value doesn't mean that `pi` itself is type `int`. For example, in the large memory model on an IBM PC using DOS, type `int` is a 2-byte value, whereas the addresses are 4-byte values.

Pointers have some other interesting properties that we'll discuss as they become relevant. Meanwhile, let's look at how pointers can be used to manage runtime allocation of memory space.

### Allocating Memory with `new`

Now that you have some feel for how pointers work, let's see how they can implement that important OOP technique of allocating memory as a program runs. So far, we've initialized pointers to the addresses of variables; the variables are named memory allocated during compile time, and the pointers merely provide an alias for memory you could access directly by name anyway. The true worth of pointers comes into play when you allocate unnamed memory during runtime to hold values. In this case, pointers become the only access to that memory. In C, you could allocate memory with the library function `malloc()`. You still can do so in C++, but C++ also has a better way, the `new` operator.

Let's try out this new technique by creating unnamed, runtime storage for a type `int` value and accessing the value with a pointer. The key is the C++ `new` operator. You tell `new` for what data type you want memory; `new` finds a block of the correct size and returns the address of the block. Assign this address to a pointer, and you're in business. Here's a sample of the technique: